Android Storage
- Vold (Honeycomb 3.2)

William.L

wiliwe@gmail.com

2011-11-25

Outline

- What is Vold?
- Components
- Communication
- Vold Init Flow
- Volume State Machine
- Example : SD Card Insert
- Appendix A- An issue in Google original code
- Appendix B- How to read Microsoft NTFS format storage?
- Reference

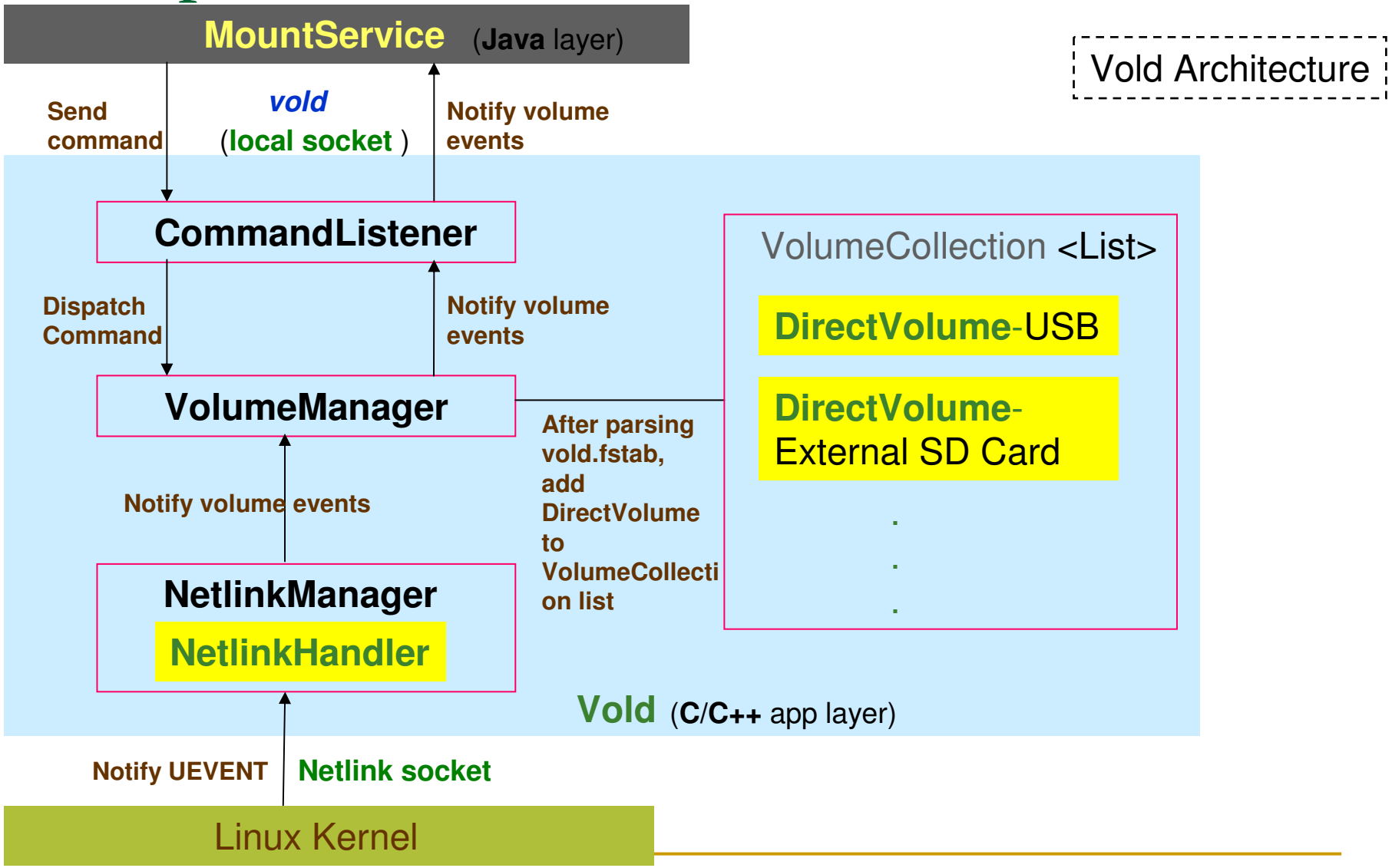
What is Vold?

- **Vol**ume **d**aemon
  - The same as **Mountd** before **Donut** (Android 1.7)
  - **Vold 2.0** since **Froyo** (Android 2.2)
  - As the role of Udev in the desktop Linux distro
  - Lisenting **Netlink** socket for volume changing Uevent
  - Ineteract with **MountService** (Java layer)
    - As a slave to MountService which is the decision maker
    - Notify volume status changing events to MountService
    - Execute commands issued from MountService
    - Communicates with MountService through **unix domain(POSIX Local IPC)** socket
  - Folder : **AndroidSrc/system/vold**
  - Main Entry
    - **main.c** : ***main***()

Components (1/12)

- Vold is composed of
  - **VolumeManager**
  - **NetlinkManager**
  - **NetlinkHandler**
  - **NetlinkEvent**
  - **CommandListener**
  - **Volume / DirectVolume**
  - **vold.fstab** file
  - **Fat**
  - **fsck_msdos** , **newfs_msdos** executables

Components (2/12)



Components (3/12)

- **VolumeManager**
  - VolumeManager.cpp/.h
  - Provides corresponding command handlers called by CommandListener for ASEC/OBB/MountService commands
  - Store volume informations retrieved from **vold.fstab**
  - Notify MountService status changing of volumes

Components (4/12)

- **NetlinkManager**
  - NetlinkManager.cpp/.h
  - Open the socket listening to kernel Netlink Kobject Uevent
  - Create a **NetlinkHandler** instance with the opened socket and activate it to start to listen to the **Netlink** socket

Components (5/12)

- **NetlinkHandler**
  - NetlinkHandler.cpp/.h
  - Wait for Netlink Kobject Uevent
  - Process Uevent in **NetlinkHandler::onEvent**()
  - Call proper command handlers(defined in VolumeManager) corresponding to the "**action**" type of uevent

Components (6/12)

- **NetlinkEvent**
  - AndroidSrc/system/core/include/sysutils/NetlinkEvent.h
  - Encapsulate Uevent information
  - Passed to **NetlinkHandler::onEvent()**

Components (7/12)

- **CommandListener**
  - CommandListener.cpp/.h
  - Using socket(**vold**) to communicate with **MountService**
  - Wait for commands from **MountService**
  - Dispatch received **MountService** commands to corresponding executor
    - CommandListener::VolumeCmd::**runCommand**()

Components (8/12)

- **Volume / DirectVolume**
  - Volume.cpp/.h , DirectVolume.cpp/.h
  - **DirectVolume** is a **subclass** of **Volume**
  - To store volume informations retreiving from vold.fstab
  - **Volume** do the actual actions(mount, unmount, etc) corresponding to the commands(doMount, doUnmount, etc) from MountService

Components (9/12)

- **vold.fstab**
  - AndroidSrc/device/nvidia/ventana [nVidia solution]
  - Describe what storages will be added into system
  - Processed by **process_config()**
    - AndroidSrc/system/vold/main.cpp
  - Example:

```
dev_mount usbdrive /mnt/usbdrive auto /devices/platform/tegra-ehci.2/usb2
```

dev_mount - Means mounting devices
usbdrive - Label for the USB volume
/mnt/usbdrive - Where the USB volume will be mounted
auto - Partition number (1 based). '**auto**' for first usable partition
/devices/platform/tegra-ehci.2/usb2 - The sysfs path to external USB device

Components (10/12)

- **Fat**
  - Fat.cpp/.h
  - Validate if the filesystem of the new added volume is FAT
  - Mount the new volume whose filesystem type is FAT
  - Format specified volume's filesystem to FAT

Components (11/12)

- **fsck_msdos** exectutable
  - AndroidSrc/external/**fsck_msdos**/**main.c**
  - To validate if the filesystem in new adding volume is **MS-DOS FAT 16 or 32**
  - Called by **Fat::check()**

Components (12/12)

- **newfs_msdos** exectutable
  - AndroidSrc/system/core/toolbox/**newfs_msdos.c**
  - To format a specified volume to **MS-DOS FAT 32** type filesystem
  - Called by **Fat::format()**

Communication

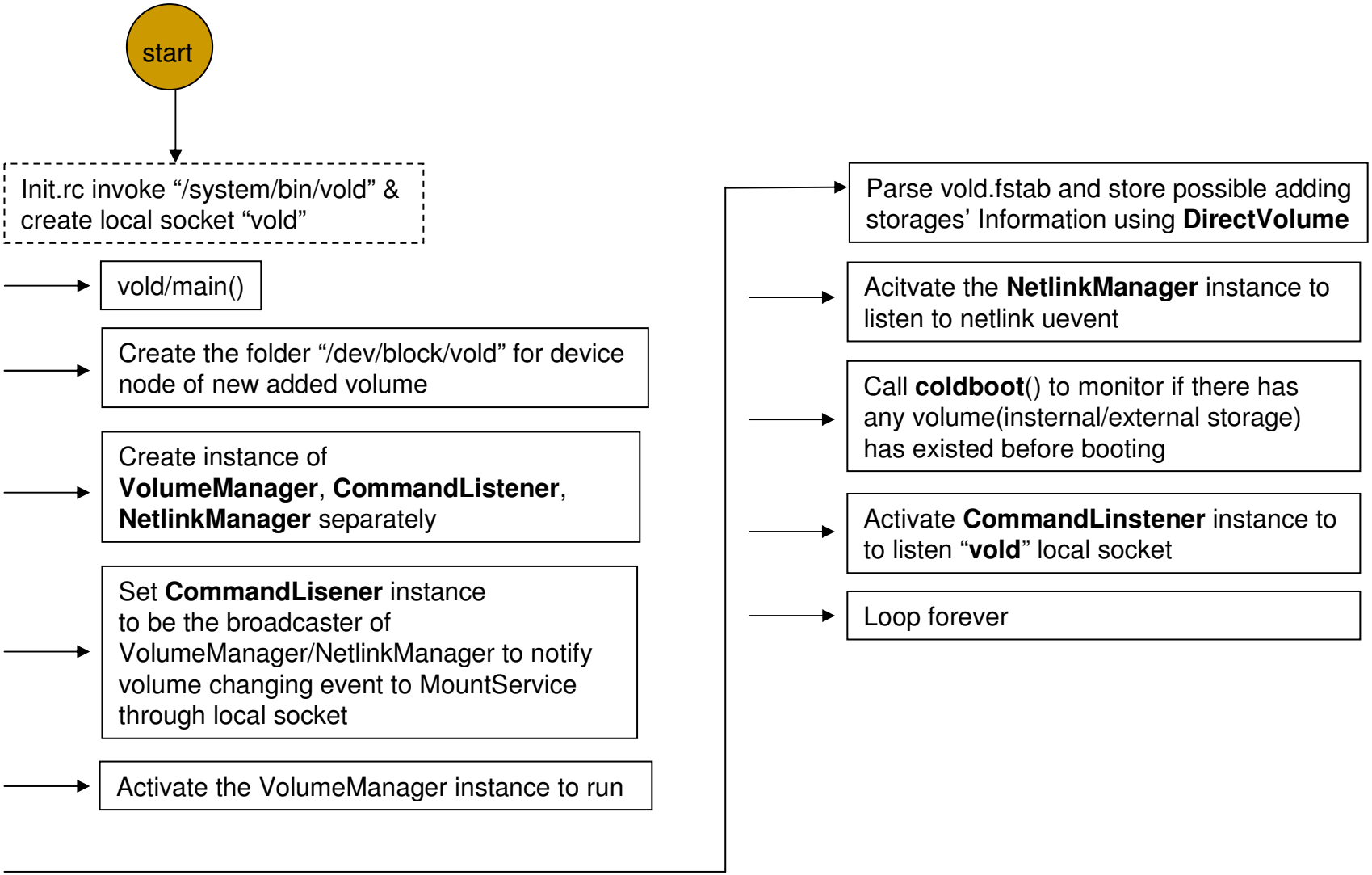
- **Netlink Socket**
  - Passes informations between **Kernel** space and **User** space
  - Protocol : **NETLINK_KOBJECT_UEVENT**
    - Defined in **KernelSrc/include/linux/netlink.h**
  - KernelSrc/lib/**kobject_uevent.c** : **kobject_uevent_env**()
- **Unix Domain/POSIX Local IPC Socket**
  - For the communication between Vold & MountService
  - Created in init.rc
    - socket **vold** stream 0660 root mount
    - mount is MountService process

AndroidSrc/system/core/rootdir/init.rc

```
service vold /system/bin/vold
    class core
    socket vold stream 0660 root mount
    ioprio be 2
```

```
# pwd
/dev/socket
# ls -ls
total 0
srw-rw---- bluetooth bluetooth          2009-01-23 18:31 dbus
srw-rw---- root      inet               2009-01-23 18:31 dnsproxyd
srwxrwxrwx root      root               2009-01-23 18:31 gps
srw------- system    system             2009-01-23 18:31 installd
srw-rw-rw- root      root               2009-01-23 18:31 keystore
srw-rw---- root      system             2009-01-23 18:31 netd
srw-rw---- root      mount              2009-01-23 18:31 vold
```

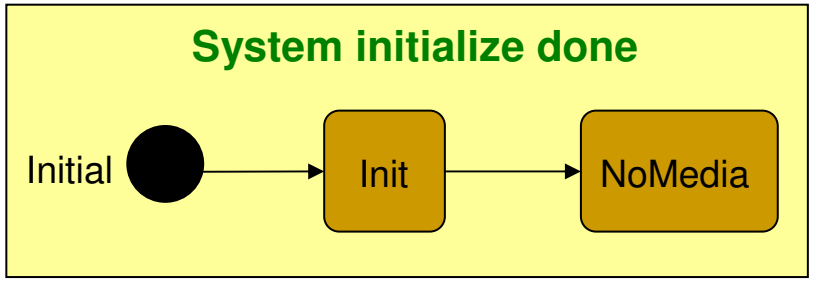
Vold Init Flow

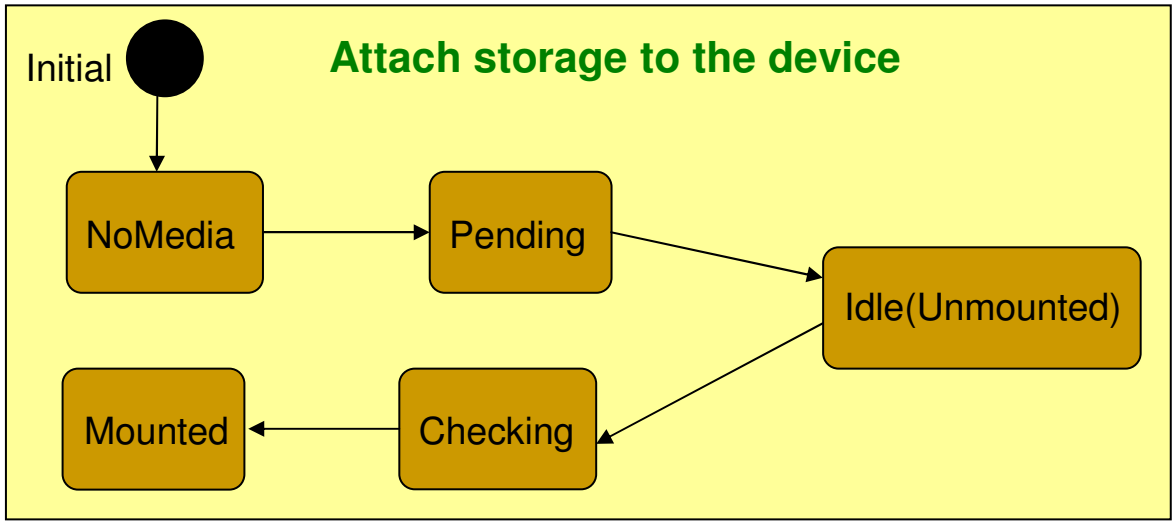


Volume State Machine (1/4)

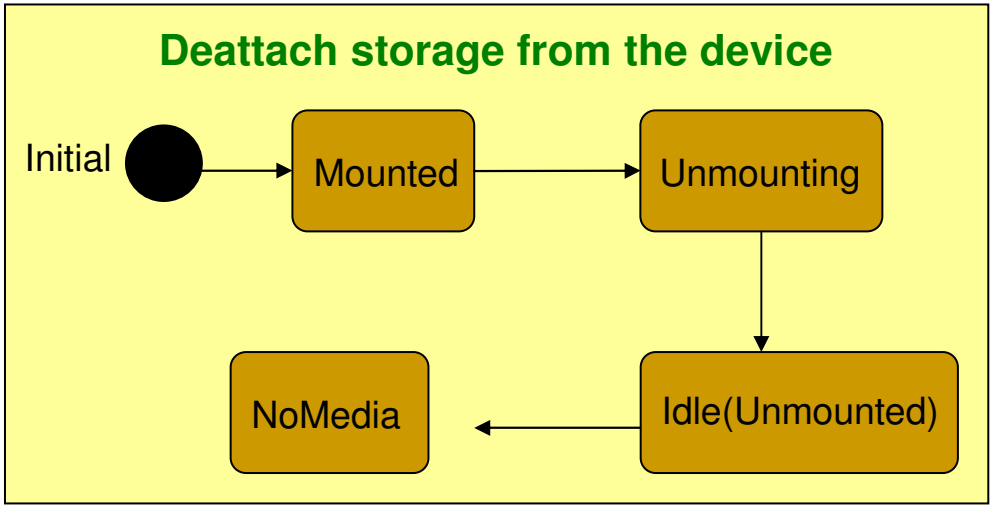
- 10 volume states (enum-value)
  - Init (-1)
  - NoMedia/Removed (0)
  - Idle/Unmounted (1)
  - Pending (2)
    - Waiting for the report of the number of volume partition
  - Chencking (3)
    - Checking filesystem type
  - Mounted (4)
  - Unmounting (5)
  - Formatting (6)
  - Shared(Unmounted) (7)
  - ShareMnt (8)

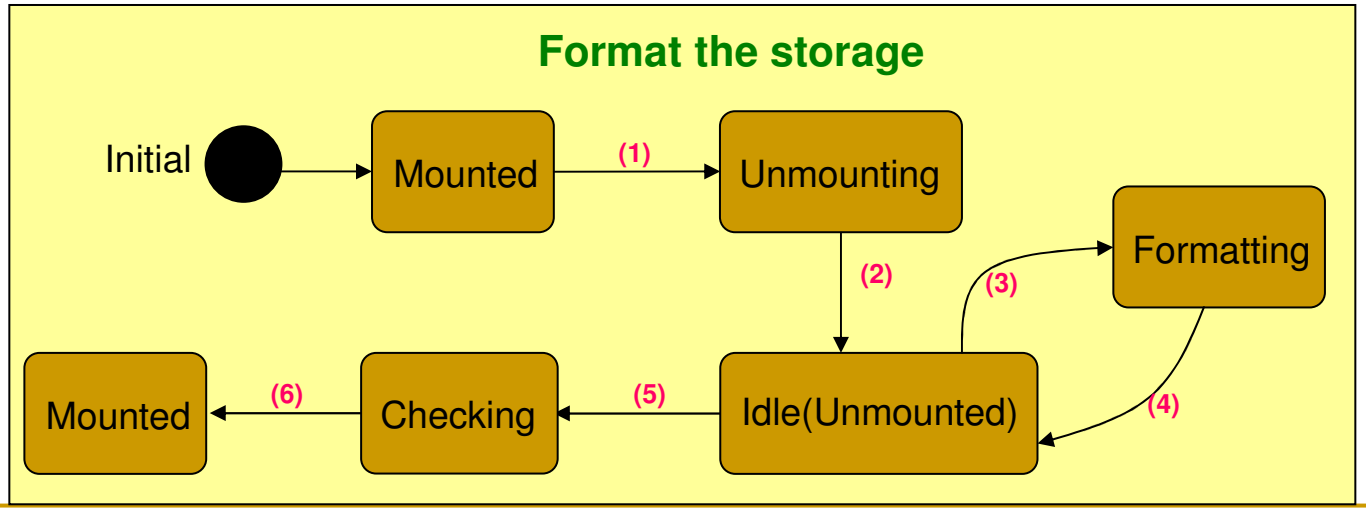
Volume State Machine (2/4)



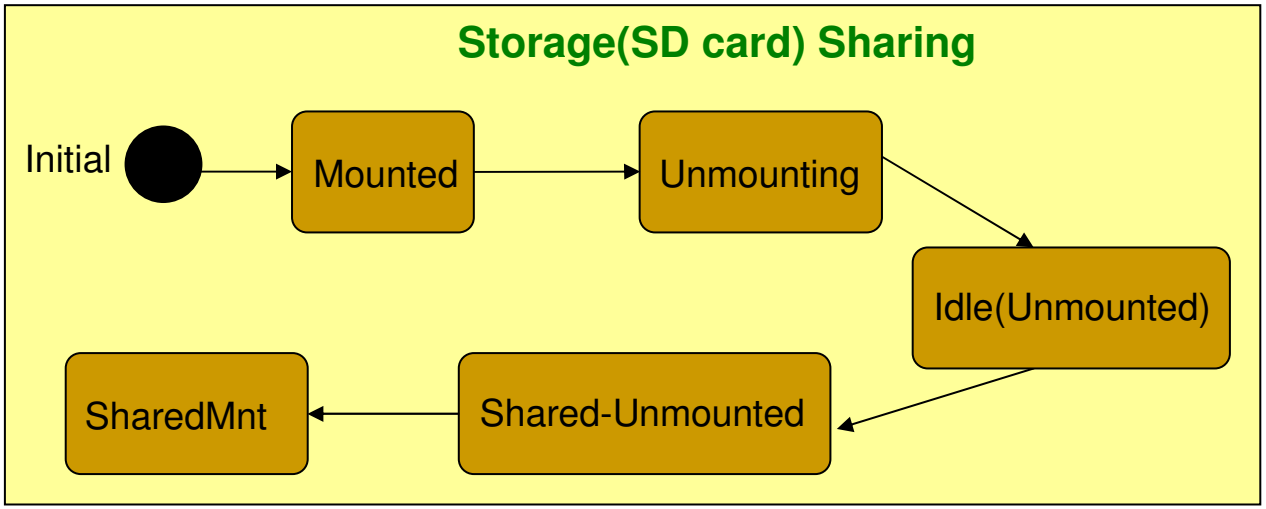


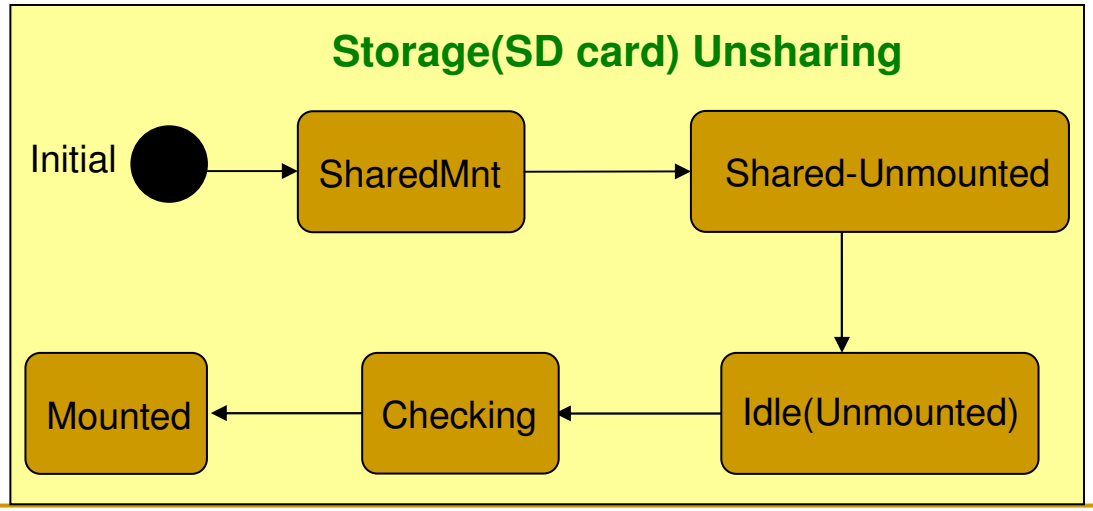
Volume State Machine (3/4)





Volume State Machine (4/4)





From Honeycomb with MTP, these two cases will not happen!

When access USB mass-storage through MTP, it need not to unmount SD card.

Example: SD Card Insert (1/3)

[Honeycomb MR2]
Volume information for SD card in **vold.fstab** :
dev_mount sdcard /sdcard2 auto /devices/platform/sdhci-tegra.2/mmc_host/mmc1

When Vold init, a **DirectVolume** instance will be created to store SD card volume informations

The stored informations are:

Label : sdcard
MountPoint : /sdcard2
DevPath(sysfs) : /devices/platform/sdhci-tegra.2/mmc_host/mmc1

Example: SD Card Insert (2/3)

Kernel detect a mmc medium (sd card) insert

Kernel send Netlink Kobject Uevent :
ACTION : add
DEVPATH : /devices/platform/sdhci-tegra.2/mmc_host/mmc1/mmc1:e624/block/mmcblk1
DEVTYPE : disk
SUBSYS : block

ACTION : add
DEVPATH : /devices/platform/sdhci-tegra.2/mmc_host/mmc1/mmc1:e624/block/mmcblk1/mmcblk1p1
DEVTYPE : partition
SUBSYS : block

NetlinkHanlder detect Uevent with "**block**" subsys type, then call **VolumeManager::handleBlockEvent**()

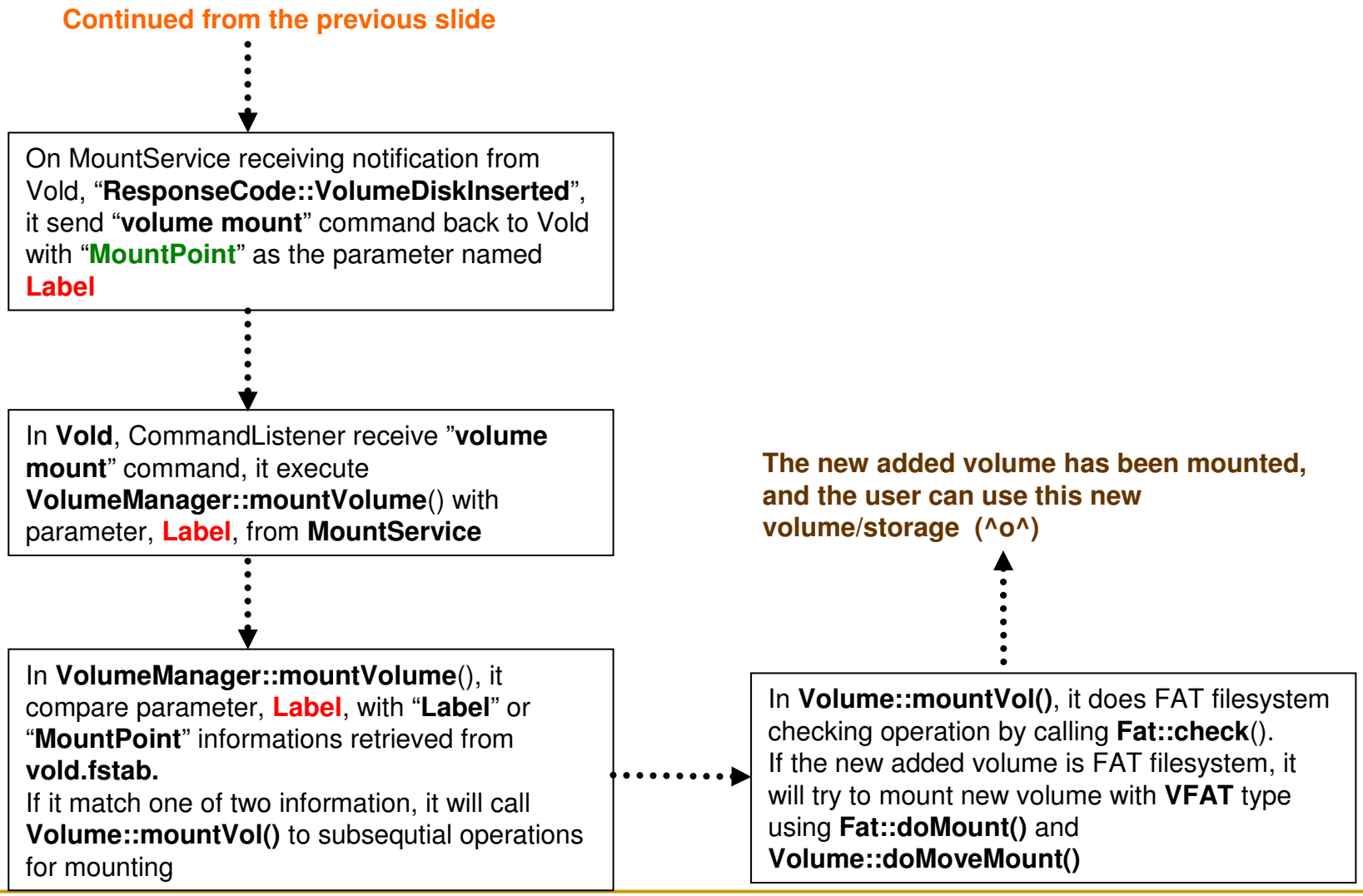
In **VolumeManager::handleBlockEvent**(), it iterate all stored DirectVolume instances and call their **handleBolckEvent**() method

In **DirectVolume::handleBolckEvent**(), it compares passed in devPath with the one retrieved from vold.fstab.

In this example, if it found the string "/devices/platform/sdhci-tegra.2/mmc_host/mmc1" in devPath, it will create a device node under **/dev/block/vold/MajrID:MinrID** for the new added volume and then send notification, "**ResponseCode::VolumeDiskInserted**", to **MountService** with **Label & MountPoint** (from vold.fstab) as parameters

Continued to the next slide

Example: SD Card Insert (3/3)



Appendix A-

An issue in Google original code:

video/image files in external storage will be deleted when un-mounting external storage

2011-10-14

Problem Description

- After user pressing "**Unmount**" button in **Settings**>**Storage** menu to unmount an external storage, all video/image files on that external storage will be deleted immediately without any warning message
- This problem will shock users!!!

Root Cause (1/2)

- After user pressing "**Unmount**" button in **Settings**>**Storage** menu, the system will send **ACTION_MEDIA_EJECT** and **ACTION_MEDIA_UNMOUNTED** Intents (events) in sequence
- When the member "**mUnmountReceiver**"

  (BroadcastReceiver) of the class **MediaProvider** (MediaProvider.java) receiving the **ACTION_MEDIA_EJECT** Intent, it will call **delete()** of **SQLiteDatabase** class to prune/remove invalid entries in thumbnail table representing video/image files on the unmounted external storage

Root Cause (2/2)

- **delete()** of **SQLiteDatabase** class will delete the (video/image) files corresponding to invalid entries in the thumbnail table eventually
- Please refer to the Google Android issue
  - http://code.google.com/p/android/issues/detail?id=3692
  - Search the keyword "**delete()**"

Solution

- To change the Intent type the member "**mUnmountReceiver**“ monitors to **ACTION_MEDIA_UNMOUNTED** and **ACTION_MEDIA_BAD_REMOVAL**
- So **MediaProvider** will prune/remove invalid entries in thumbnail table only when it receives **Unmounted** or **Bad-Removal** events
  - e.g. the external storage has been detached from the file system actually!

Appendix B-
How to read Microsoft NTFS format storage?

2011-12-27

Problem Description

- Originally, Google Android only supports **Microsoft FAT** format storage
- **Android Vold** (Volume.cpp) checks the format of a new inserted storage when trying to do mounting. If it is not FAT format, it will skip this storage and check the next found storage
- Storages in NTFS format are so common, so it is better for an Android device to be capable of reading NTFS format storage

Solution (1/2)

- **TuxEra**'s **NTFS-3G** is a stable, full-featured solution to address the requirement of reading NTFS format storage
  - This is a file system driver for NTFS format
  - After installing this into Linux system, it will add a file system type option "**ntfs**" of "**mount**" system command and "**mount()**" API
- **NTFS-3G** official site
  - **http://www.tuxera.com/community/ntfs-3g-download/**

Solution (2/2)

- In the method **mountVol()** of **Volume** class (**Volume.cpp**)
  - "**if (Fat::check(devicePath))**" block will enter when the checking status is non-zero, e.g. the current storage is **NOT FAT** format
  - It could add codes to calling **mount()** with "**ntfs**" option **in this block** to try to mount current storage and to see if it is **NTFS** format through the return value of **mount()**
  - Also, it needs to check the storage is in **Read-Only** mode

Reference

- **Kobject Uevent socket** – Chapter **17**, Linux 核心開發指南 (Linux Kernel Development, 2/e) Robert Love 著, 沈中庸、沈彥男 譯, 維科出版社

- **Unix Domain Socket** – Chapter **15**, Unix Network Programming, Volume 1: The Sockets Networking API, 3/e W. Richard Stevens

- Netlink in Kernel – http://www.linuxjournal.com/article/7356